Zeno: Byzantine-suspicious stochastic gradient descent

Cong Xie
Department of Computer Science
University of Illinois at Urbana Champaign
`cx2@illinois.edu`

Oluwasanmi Koyejo
Department of Computer Science
University of Illinois at Urbana Champaign
`cx2@illinois.edu`

Indranil Gupta
Department of Computer Science
University of Illinois at Urbana Champaign
`cx2@illinois.edu`

Abstract

We propose Zeno, a new robust aggregation rule, for distributed synchronous Stochastic Gradient Descent (SGD) under a general Byzantine failure model. The key idea is to suspect the workers that are potentially malicious, and use a ranking-based preference mechanism. This allows us to generalize beyond past work–in our case, the number of malicious workers can be arbitrarily large, and we use only the weakest assumption on honest workers (at least one honest worker). We prove the convergence of SGD under these scenarios. Empirical results show that Zeno outperforms existing approaches under various attacks.

1 Introduction

Distributed machine learning is widely used for accelerating the model training. As the number of workers gets larger, the distributed system becomes more vulnerable to various kinds of failures or even attacks [7]. For this reason, fault tolerance in distributed settings has attracted increasing attention in the machine-learning community [1, 2, 4, 6, 17, 18, 19, 20]. In certain distributed settings, such as federated learning [9, 10] and volunteer computing [14, 16], the workers/contributors can be unreliable. The computation nodes can be personal mobile devices or denoted personal computers/workstations. The attackers and unreliable workers can upload poisonous updates to the aggregated machine-learning models. Thus, failure/attack resilience in distributed machine-learning systems is now becoming increasingly important.

We study the fault tolerance of the Parameter Server (PS) architecture for distributed synchronous stochastic gradient descent (SGD). In the PS architecture, the processes are composed of the server nodes and the worker nodes. In each SGD iteration, the workers pull the latest model from the servers, estimate the gradients using the local portion of the training data, and send the gradient estimators to the servers. The servers aggregate the gradient estimators, update the model by using the aggregated gradients, and broadcast the latest model to the workers.

Based on the PS architecture, we consider the most general failure model, namely Byzantine failures [12]. Here the malicious workers know the entire system, and are allowed to behave arbitrarily. Typically, the number of Byzantine workers is assumed to be smaller than the number of non-Byzantine ones [1, 2, 4, 17, 19, 20]. In contrast, we require only the weakest assumption that there is at least one non-Byzantine worker. Furthermore, the Byzantine workers can collude. The group of Byzantine gradients can behave similarly to the correct gradients in the variance, and magnitude, which makes them less distinguishable from the correct ones.

Preprint. Work in progress.

We study the Byzantine resilience of synchronous SGD with weak assumptions on the number of Byzantine workers. To achieve this flexibility, we treat each candidate gradient estimator as a suspect. We compute a score using a stochastic first order oracle which indicates how trustworthy the given worker is. Then, we take the average over the several candidates with the highest scores. In summary, the main contributions of this paper are listed below:

- We propose a new aggregation rule for synchronous SGD with provable convergence when there is at least one non-Byzantine worker. As far as we know, this paper is the first to theoretically and empirically study the cases where there are more Byzantine workers than non-Byzantine ones.
- We show experimentally that the existing majority-based algorithms may fail, even when there are fewer Byzantine workers than non-Byzantine ones. These results elucidate the failure modes of recent published approaches. Furthermore, we show that our suspicion-based algorithm can handle such cases.

2 Related work

The robust estimator of the mean is a well-studied topic in statistics [8]. The idea of robust statistics is also applied to machine learning in Byzantine settings. For example, Chen et al. [4], Su and Vaidya [17, 18] use geometric median as the aggregation rule. Yin et al. [20] establishes statistical error rates for marginal trimmed mean as the aggregation rule.

There are also robust gradient aggregation rules that are not based on robust statistics. For example, Blanchard et al. [2], Mhamdi et al. [15] propose Krum and its variants, which select the candidates with minimal local sum of Euclidean distances.

Alistarh et al. [1] propose a Byzantine-resilient SGD variant different from the robust aggregation rules. The algorithm utilizes the historical information, and achieves the optimal sample complexity. However, the algorithm requires the estimated upper bound of the variances of the stochastic gradients, which makes the algorithm less practical. Furthermore, there are no empirical results provided.

Damaskinos et al. [5] consider a different scenario, where the distributed training is asynchronous. The proposed algorithm, Kardam, utilizes the Lipschitzness of the loss function to filter out the outliers. However, such an algorithm still relies on the majority rule, which requires the number of Byzantine workers less than one-third of the total number of workers.

In summary, the existing majority-based methods for synchronous SGD [1, 2, 4, 17, 19, 20] assume that the non-Byzantine workers dominates the entire set of workers. Thus, such algorithms can trim the outliers from the candidates. However, in real-world attacks, there are no guarantees that the number of Byzantine workers can be bounded from above.

3 Model

We consider the following optimization problem:

$$\min_x F(x),$$

where $F(x) = \mathbb{E}_{z \sim \mathcal{D}}[f(x; z)]$, z is sampled from some unknown distribution $\mathcal{D}$. We assume that there exists a minimizer of $F(x)$, which is denoted by x^*.

We solve this problem in a distributed manner with m workers. In each iteration, each worker will sample n independent and identically distributed (i.i.d.) data points from the distribution $\mathcal{D}$, and compute the gradient of the local empirical loss $F_i(x) = \frac{1}{n} \sum_{j=1}^{n} f(x; z^{i,j}), \forall i \in [m]$, where $z^{i,j}$ is the jth sampled data on the ith worker. The servers will collect and aggregate the gradients sent by the works, and update the model as follows:

$$x^{t+1} = x^t - \gamma^t \texttt{Aggr}(\{g_i(x^t) : i \in [m]\}),$$

where $\texttt{Aggr}(\cdot)$ is an aggregation rule (e.g., averaging), and

$$g_i(x^t) = \begin{cases} * & i\text{th worker is Byzantine,} \\ \nabla F_i(x^t) & \text{otherwise.} \end{cases} \tag{1}$$

Formally, we define the Byzantine failures in synchronous SGD as follows.

Definition 1. *(Byzantine Failures). In the tth iteration, let $\{v_i^t : i \in [m]\}$ be i.i.d. random vectors in $\mathbb{R}^d$, where $v_i^t = \nabla F_i(x^t)$. The set of correct vectors $\{v_i : i \in [m]\}$ is partially replaced by Byzantine vectors, which results in $\{\tilde{v}_i^t : i \in [m]\}$, where $\tilde{v}_i^t = g_i(x^t)$ as defined in Equation (1). In other words, a correct/non-Byzantine gradient is $\nabla F_i(x^t)$, while a Byzantine gradient, marked as "∗", is assigned arbitrary value. We assume that q out of m vectors are Byzantine, where $q < m$. Furthermore, the indices of Byzantine workers can change across different iterations.*

To help understand the Byzantine failures in synchronous SGD, we illustrate a toy example in Figure 1.
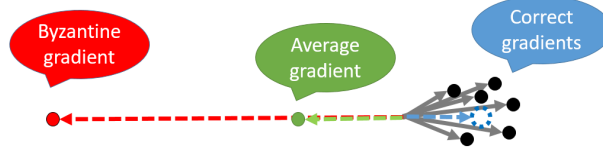

Figure 1: A toy example of the Byzantine failures in synchronous SGD. There are totally $m = 7$ candidate gradient estimators. The black dots represent the correct gradients, where $\tilde{v}_i = \nabla F_i(x^t)$, $i \in [m-1]$. The red dot represents the Byzantine gradient, whose value is $\tilde{v}_m = \epsilon \nabla F_m(x^t)$, where $\epsilon < 0$ is a large negative constant. The blue dashed circle represent the expectation of the true gradient $\nabla F(x^t)$. Thus, the averaged gradient, which will be computed by the server, represented by the green dot, is far away from the true gradient, which is harmful to the model training.

4   Methodology

In contrast to the existing majority-based methods, we compute a score for each candidate gradient estimator by using the stochastic first-order oracle. We rank each candidate gradient estimator based on the estimated descent of the loss function, and the magnitudes. Then, the algorithm aggregates the candidates with highest scores. The score roughly indicates how trustworthy each candidate is.

Definition 2. *(Stochastic Descendant Score) Denote $f_r(x) = \frac{1}{n_r} \sum_{i=1}^{n_r} f(x; z_i)$, where z_i's are i.i.d. samples drawn from $\mathcal{D}$, and n_r is the batch size of $f_r(\cdot)$. $\mathbb{E}[f_r(x)] = F(x)$. For any update (gradient estimator) u, based on the current parameter x, learning rate γ, and a constant weight $\rho > 0$, we define its stochastic descendant score as follows:*

$$Score_{\gamma,\rho}(u, x) = f_r(x) - f_r(x - \gamma u) - \rho \|u\|^2.$$

The score defined in Definition 2 is composed of two parts: the estimated descendant of the loss function, and the magnitude of the update. The score increases when the estimated descendant of the loss function, $f_r(x) - f_r(x - \gamma \tilde{v}_i)$, increases. The score decreases when the magnitude of the update, $\|\tilde{v}_i\|^2$, increases. Intuitively, the larger descendant suggests faster convergence, and the smaller magnitude suggests a smaller step size. Even if a gradient is Byzantine, a smaller step size makes it less harmful and easier to be cancelled by the correct gradients.

Using the score defined above, we establish the following suspicion-based aggregation rule. We ignore the index of iterations, t, for convenience.

Definition 3. *(Suspicion-based Aggregation) Assume that among the gradient estimators $\{\tilde{v}_i : i \in [m]\}$, q elements are Byzantine, and x is the current value of the parameters. We sort the sequence by using the stochastic descendant score defined in Definition 2, which results in $\{\tilde{v}_{(i)} : i \in [m]\}$, where*

$$Score_{\gamma,\rho}(\tilde{v}_{(1)}, x) \geq Score_{\gamma,\rho}(\tilde{v}_{(2)}, x) \geq \ldots \geq Score_{\gamma,\rho}(\tilde{v}_{(m)}, x).$$

In other words, $\tilde{v}_{(i)}$ is the vector with the ith highest score in $\{\tilde{v}_i : i \in [m]\}$.

The proposed aggregation rule, Zeno[1], aggregates the gradient estimators by taking the average of the first $m - b$ elements in $\{\tilde{v}_{(i)} : i \in [m]\}$ (the gradient estimators with the $(m - b)$ highest scores):

$$\texttt{Zeno}_b(\{\tilde{v}_i : i \in [m]\}) = \frac{1}{m - b} \sum_{i=1}^{m-b} \tilde{v}_{(i)},$$

---

[1]The name of a Byzantine emperor.

where $m > b \geq q$.

Note that z_i's (in Definition 2) are independently sampled in different iterations. Furthermore, in each iteration, as z_i's are sampled after the arrival of the candidate gradient estimators $\tilde{v}_i^t$, the Byzantine workers cannot obtain the information of $f_r(\cdot)$, which means that the Byzantine gradients are independent of $f_r(\cdot)$.

5 Theoretical guarantees

In this section, we prove the convergence of synchronous SGD with Zeno as the aggregation rule under Byzantine failures. We start with the assumptions required by the convergence guarantees. The two basic assumptions are the smoothness of the loss function, and the bounded variance of the (non-Byzantine) gradient estimators.

5.1 Assumptions

Assumption 1. *(Bounded Taylor's Approximation) We assume that $f(x; z)$ has L-smoothness and μ-lower-bounded Taylor's approximation:*

$$\langle \nabla f(x; z), y - x \rangle + \frac{\mu}{2} \|y - x\|^2 \leq f(y; z) - f(x; z) \leq \langle \nabla f(x; z), y - x \rangle + \frac{L}{2} \|y - x\|^2,$$

where $\mu \leq L$, and $L > 0$.

Note that Assumption 1 covers the case of non-convexity by taking $\mu < 0$, non-strong convexity by taking $\mu = 0$, and strong convexity by taking $\mu > 0$.

Assumption 2. *(Bounded Variance) We assume that in any iteration, any correct gradient estimator $v_i = \nabla F_i(x)$ with $\mathbb{E}[v_i] = \nabla F(x)$ has the upper-bounded variance: $\mathbb{E}\|v_i - \nabla F(x)\|^2 \leq V$. Furthermore, for simplicity, we assume that there exists a constant α, which is large enough such that $\mathbb{E}\|v_i\|^2 \leq \alpha V$. And, for the stochastic descent score, we assume that $\mathbb{E}\|\nabla f_r(x) - \nabla F(x)\|^2 \leq V_r, \forall x \in \mathbb{R}^d$.*

In general, Assumption 2 bounds the variance and the second-order moment of the correct gradients of any sample loss function $f(x; z)$, $z \sim \mathcal{D}$.

5.2 Convergence guarantees

We first bound the variance of the aggregated vector in the following theorem. Note that we ignore the index of iteration t for convenience. The proof can be found in the appendix.

Theorem 1. *(Bounded Variance) Assume that q out of the m candidate gradient estimators $\{\tilde{v}_i : i \in [m]\}$ are Byzantine, with $q \leq b < m$, and x is the current value of the parameters. We further assume that ρ is large enough such that $\mu\gamma + \frac{2\rho}{\gamma} \geq 1$. By averaging the first $m - b$ vectors in $\{\tilde{v}_{(i)} : i \in [m]\}$, the aggregated vector $\texttt{Zeno}_b(\{\tilde{v}_i : i \in [m]\})$ is bounded by*

$$\mathbb{E} \|\texttt{Zeno}_b(\{\tilde{v}_i : i \in [m]\}) - \nabla F(x)\|^2 \leq \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2},$$

where $m > b \geq q$, and $\beta = \alpha \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)$.

Note that Theorem 1 tells us that the variance decreases when m increases, b decreases, β decreases, V decreases, or V_r decreases. This bound also tells us that when q approaches 0, the term $\frac{8q(m-q)\beta V_r}{(m-b)^2}$ is also close to 0. However, when q is large, we need to decrease V_r to stabilize the convergence, which potentially requires more computation cost with larger batch size of n_r for evaluation.

Using the bounded variance above, we establish the convergence guarantees for synchronous SGD with Zeno as the aggregation rule.

5.2.1 Strongly convex functions

We prove linear convergence with constant error for strongly convex loss functions using Zeno as the aggregation rule. The proof can be found in the appendix.

Theorem 2. *Assume that $F(x)$ is μ_F-strongly convex and L_F-smooth, where $0 < \mu_F \leq L_F$. We take $\gamma \leq \frac{2}{\mu_F + L_F}$. Using* Zeno$(\cdot)$ *and taking the same assumptions in Theorem 1, we obtain linear convergence with a constant error after T iterations:*

$$\mathbb{E}\|x^T - x^*\| \leq \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F}\right)^T \|x^0 - x^*\| + \frac{\mu_F + L_F}{\mu_F L_F} \gamma \sqrt{\Delta},$$

where $\Delta = \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}$, $\beta = \alpha \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)$.

In practice, we usually take $\rho = c\gamma$, where c is a positive constant which makes ρ proportional to the learning rate γ. Furthermore, if we use a diminishing learning rate, the error $\gamma \sqrt{\Delta}$ will be reduced.

5.2.2 General functions

For general non-strongly convex functions and non-convex functions, we provide the following convergence guarantee. The proof can be found in the appendix.

Theorem 3. *Assume that $F(x)$ is L_F-smooth and potentially non-convex, where $0 < L_F$. We take $\gamma \leq \frac{1}{L_F}$. Using* Zeno$(\cdot)$ *and taking the same assumptions in Theorem 1, we obtain the convergence with a constant error after T iterations:*

$$\frac{\sum_{i=0}^{T-1} \mathbb{E}\|\nabla F(x^i)\|^2}{T} \leq \frac{2}{\gamma T} \left[F(x^0) - F(x^*)\right] + \Delta,$$

where $\Delta = \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}$, $\beta = \alpha \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)$.

Remark 1. *There are two practical concerns for the proposed algorithm. First, by increasing the batch size of $f_r(\cdot)$ (n_r in Definition 2), the error can be reduced. However, larger n_r also requires more computation time, which slows down the aggregation. Second, theoretically we need larger ρ for non-convex problems. However, larger ρ makes* Zeno *less sensitive to the descendant of the loss function, which potentially increases the risk of aggregating harmful candidates. In practice, we can use a small ρ by assuming the local convexity of the loss functions.*

6 Experiments

In this section, we evaluate the Byzantine resilience of the proposed algorithm. We summarize our results here:

- Compared to the baselines, Zeno shows better convergence when there are more Byzantine workers than non-Byzantine ones.
- Compared to the baselines, Zeno shows better convergence when the Byzantine gradients do not change the magnitude, which makes them less distinguishable from the correct ones.
- Zeno is robust to the choices of the hyperparameters, including the Zeno batch size n_r, the weight ρ, and the number of trimmed elements b.

6.1 Datasets and evaluation metrics

We conduct experiments on two image classification tasks: handwritten digits classification on MNIST dataset and object recognition on CIFAR-10 dataset. On the MNIST dataset, we use softmax regression for convex optimization, and multi-layer perceptron (MLP) for non-convex optimization. On the CIFAR-10 dataset, we use convolutional neural network (CNN). The details of the datasets are listed in Table 1. The details of the neural network structures can be found in the appendix. Due to the space limitation, we only show the results on MNIST dataset with MLP. The additional experiments, including MNIST with softmax regression and CIFAR-10 with CNN, can be found in the appendix.

In each experiment, we launch twenty worker processes. We repeat each experiment ten times and take the average. We use top-1 accuracy on the test sets as the evaluation metric.

6.1.1 Baselines

We use the averaging without failures/attacks as the gold standard, which is referred to as `Mean without Byzantine`. Note that this method is not affected by b or q. The baseline aggregation rules are `Mean`, `Median`, and `Krum` as defined below.

Definition 4. *(Median [19, 20]) We define the marginal median aggregation rule* `Median`$(\cdot)$ *as*

$$med = \texttt{Median}(\{\tilde{v}_i : i \in [m]\}),$$

where for any $j \in [d]$, the jth dimension of med is $med_j = median\left(\{(\tilde{v}_1)_j, \ldots, (\tilde{v}_m)_j\}\right)$, $(\tilde{v}_i)_j$ is the jth dimension of the vector $\tilde{v}_i$, $median(\cdot)$ is the one-dimensional median.

Definition 5. *(Krum [2])*

$$\texttt{Krum}(\{\tilde{v}_i : i \in [m]\}) = \tilde{v}_k, \quad k = \underset{i \in [m]}{\operatorname{argmin}} \sum_{i \to j} \|\tilde{v}_i - \tilde{v}_j\|^2,$$

where $i \to j$ is the indices of the $m - q - 2$ nearest neighbours of $\tilde{v}_i$ in $\{\tilde{v}_i : i \in [m]\}$ measured by Euclidean distances.

6.2 Sign-flipping attack

In this section, we test the Byzantine resilience to the sign-flipping attack, which flips the sign of each victim gradient and enlarges the magnitude. To be more specific, each gradient, if chosen to be Byzantine, will be multiplied by ϵ, which results in $\tilde{v}_i = \epsilon v_i$, where $\epsilon \leq -1$. We conduct the experiments on MNIST dataset with MLP. The results are shown in Figure 2. We can see that `Zeno` outperforms the other aggregation rules. When q is small, `Zeno` and `Krum` perform similarly. With large $|\epsilon|$, `Zeno` behaves as if there is no Byzantine failures. Surprisingly, `Krum` performance well with large q and $|\epsilon|$. We explain this phenomenon further in Section 6.5.

6.3 Omniscient attack

In this section, we test the Byzantine resilience to the omniscient attack. Such attack sets each Byzantine gradient as $\tilde{v}_i = \epsilon \frac{1}{m} \sum_{i=1}^{m} v_i$, where $\epsilon \leq -1$. We conduct the experiments on MNIST dataset with MLP. The results are shown in Figure 3. We can see that `Zeno` outperforms the other aggregation rules. Furthermore, `Zeno` performs much better than the baselines when there are more Byzantine workers than honest workers ($q = 12$). Smaller $|\epsilon|$ slows down the convergence of `Zeno`. Interestingly, averaging performs well with small q and $|\epsilon|$, while `Krum` diverges with large $|\epsilon|$ even if $2q \leq m - 2$. We explain this phenomenon further in Section 6.5.

6.4 Hyperparameter sensitivity

In this section, we tune the hyperparameters of `Zeno`, and show how they affect the performance under the sign-flipping attack. The hyperparameters include the batch size n_r for evaluating the stochastic descendant score in Definition 2, the weight ρ, and the number of trimmed workers b. The results are shown in Figure 4. It is shown that larger n_r, smaller ρ, larger b and smaller q improve the accuracy. Furthermore, `Zeno` is generally robust to different n_r, b, and q. However, when ρ is too large, with large q `Zeno` can fail.

6.5 Discussion

The experimental results show some interesting heuristics for both attackers and defenders:

Table 1: Dataset summary

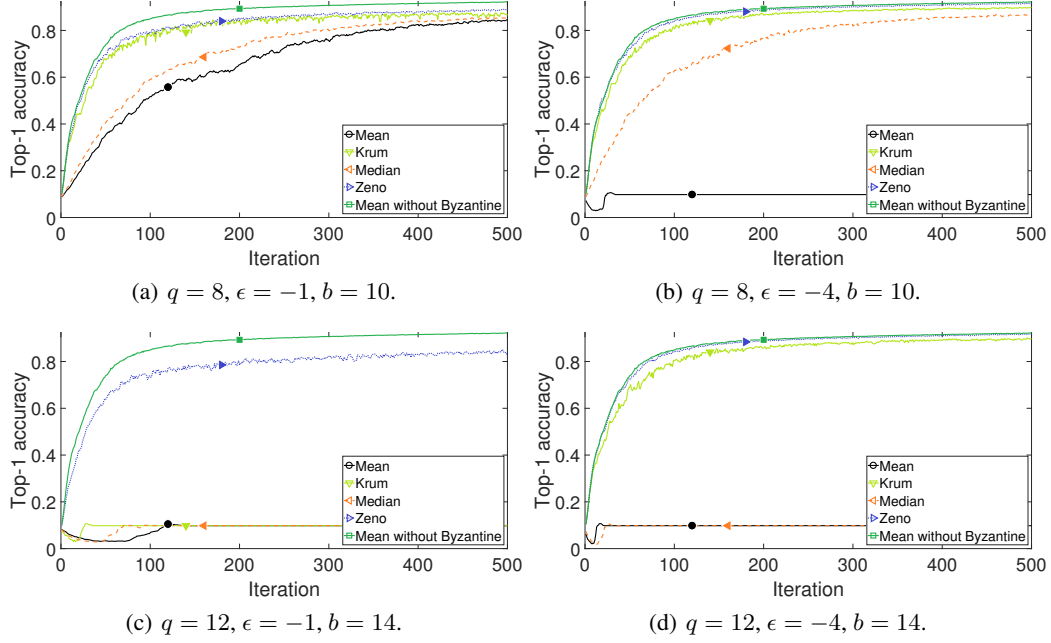| Dataset	# train	# test	# rounds	Evaluation metric
MNIST [13]	60k	10k	500	top-1 accuracy on the test set
CIFAR10 [11]	50k	10k	10000	top-1 accuracy on the test set

Figure 2: Top-1 accuracy of MLP on MNIST with sign-flipping attack and different hyperparameters. In all the 4 figures, $\gamma = 0.1$, $\rho = \frac{\gamma}{40}$, $n_r = 12$, worker batch size is 32. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.
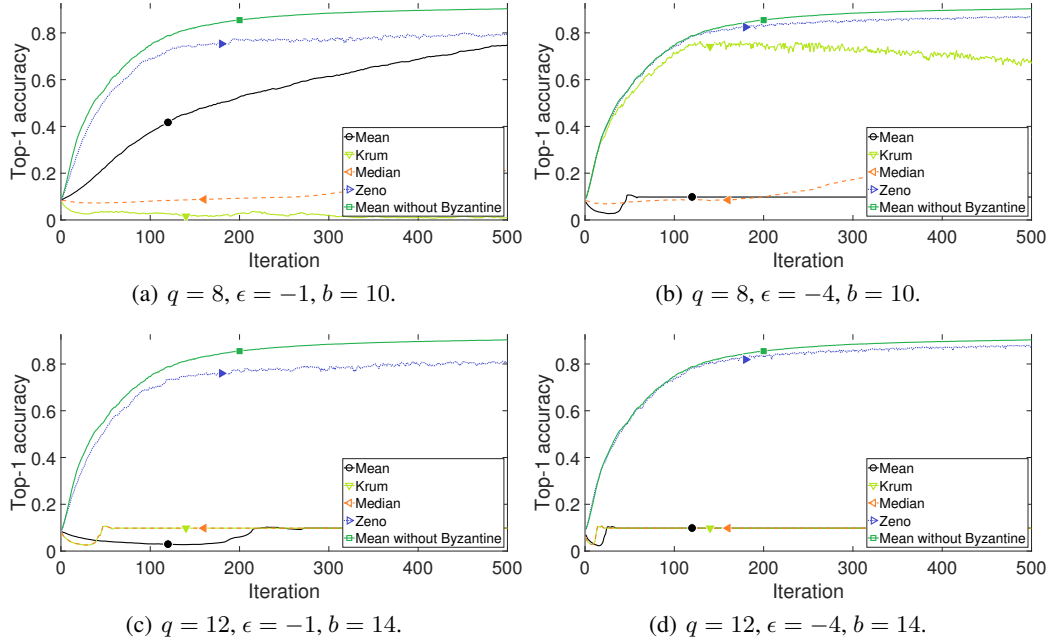

Figure 3: Top-1 accuracy of MLP on MNIST with omniscient attack and different hyperparameters. In all the 4 figures, $\gamma = 0.05$, $\rho = \frac{\gamma}{100}$, $n_r = 12$, worker batch size is 32. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.

- The unchanged magnitude ($|\epsilon| = 1$) makes the attack more harmful to distance-based algorithms. When the magnitude of the Byzantine gradients, rescaled by $|\epsilon|$, is too large or too small compared to the magnitude of the normal ones, the Byzantine gradients are more easily to be distinguished. Especially, for Krum and Zeno, larger $|\epsilon|$ improves the performance. Median, based on sorting

7

(a) $\rho = \frac{\gamma}{40}$, n_r varies.

(b) $n_r = 12$, ρ varies.

(c) $n_r = 12$, b varies.
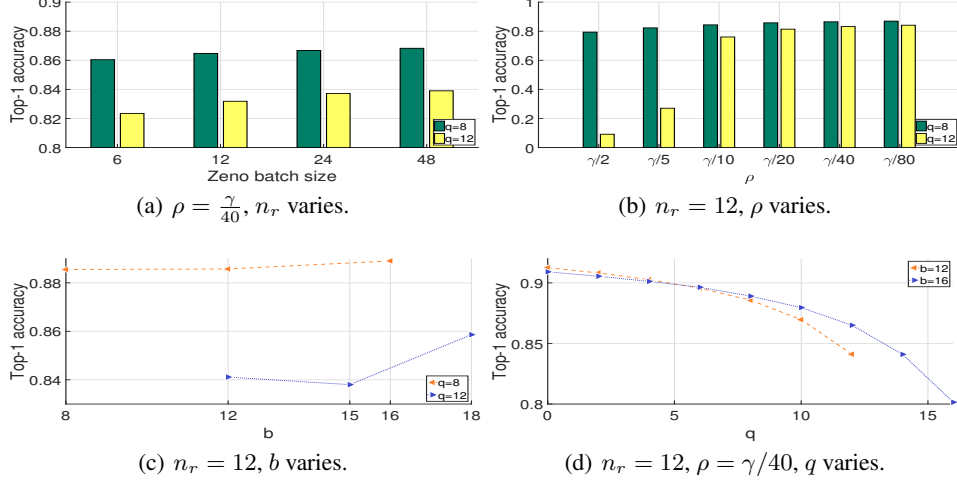
(d) $n_r = 12$, $\rho = \gamma/40$, q varies.

Figure 4: Top-1 accuracy of MLP on MNIST with sign-flipping attack and different hyperparameters. We use $\gamma = 0.05$, $\epsilon = -1$. Worker batch size is 32.

instead of distances, is not affected by the magnitude. Another exception is `Mean`. When the correct gradients dominate and $|\epsilon|$ is small enough, the sum of Byzantine gradients are not large enough to cancel the correct ones. Thus in this special case, standard aggregation actually outperforms current Byzantine-tolerant methods–a failure case that was seemingly missed by earlier methods, as shown in Figure 2(a), and 3(a).

- Collusion makes the attacks more harmful to majority-based algorithms. For sign-flipping attacks, even when the Byzantine workers dominate, with large $|\epsilon|$, `Krum` can still have good performance. The reason is that `Krum` aggregates the cluster of gradients, where the gradients inside the cluster are close to each other. When $|\epsilon|$ gets larger, the sign-flipping attack also makes the Byzantine gradients farther away from each other, which makes them filtered out by `Krum`. In contrast, under omniscient attacks, all the Byzantine gradients have the same values, which fools `Krum`. `Median`, though more robust than `Krum`, also has the same issue. Note that under omniscient attacks, even when the correct workers dominate ($q = 8$), `Krum` can diverge, as shown in Figure 3(a) and (b). Such phenomenon is not discussed in the original paper [2].

In general, we find that `Zeno` is more robust than the current state of the art. Especially, when the Byzantine workers dominate, `Zeno` is the only one that converges in all experiments. When the correct workers dominate, `Median` can be an alternative with cheap computation. The computational complexity of `Zeno` depends on the complexity of inference and the Zeno batch size n_r. These additional hyperparameters make direct comparison to standard methods more challenging.

As expected, the empirical results of hyperparameter sensitivity verify most of the theoretical analysis. Interestingly, though the upper bound of error theoretically grows with b, the empirical results show better convergence with larger b. The reason is that, with larger b, `Zeno` filters out more potentially harmful gradients, which improves the convergence. Furthermore, we find that the sensitivity to `Zeno` batch size n_r is limited, which means that small n_r is enough to obtain good convergence. Such a result also indicates that the computational complexity of `Zeno` can be reduced without influencing the convergence. `Zeno` is more sensitive to ρ, especially when the Byzantine workers dominate. Typically, we need smaller ρ for better convergence. Once ρ is small enough (e.g., smaller than $\frac{\gamma}{20}$ in Figure 4(d)), further decreasing ρ will not make significant differences.

7   Conclusion

We propose a novel aggregation rules for synchronous SGD, which takes the weakest assumption that there is at least one honest worker. The algorithm has provable convergence. Our empirical results show good performance in practice. We will apply the proposed method to asynchronous SGD in the future work.

References

[1] D. Alistarh, Z. Allen-Zhu, and J. Li. Byzantine stochastic gradient descent. *arXiv preprint arXiv:1803.08917*, 2018.

[2] P. Blanchard, R. Guerraoui, J. Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 118–128, 2017.

[3] S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[4] Y. Chen, L. Su, and J. Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *POMACS*, 1:44:1–44:25, 2017.

[5] G. Damaskinos, E. M. E. Mhamdi, R. Guerraoui, R. Patra, and M. Taziki. Asynchronous byzantine machine learning. *arXiv preprint arXiv:1802.07928*, 2018.

[6] J. Feng, H. Xu, and S. Mannor. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

[7] D. Harinath, P. Satyanarayana, and M. R. Murthy. A review on security issues and attacks in distributed systems. *Journal of Advances in Information Technology*, 8(1), 2017.

[8] P. J. Huber. Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer, 2011.

[9] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *CoRR*, abs/1610.02527, 2016.

[10] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016.

[11] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[12] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4:382–401, 1982.

[13] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.

[14] E. Meeds, R. Hendriks, S. al Faraby, M. Bruntink, and M. Welling. Mlitb: machine learning in the browser. *PeerJ Computer Science*, 1, 2015.

[15] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

[16] K. Miura and T. Harada. Implementation of a practical distributed calculation system with browsers and javascript, and application to distributed deep learning. *CoRR*, abs/1503.05743, 2015.

[17] L. Su and N. H. Vaidya. Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms. In *PODC*, 2016.

[18] L. Su and N. H. Vaidya. Defending non-bayesian learning against adversarial attacks. *arXiv preprint arXiv:1606.08883*, 2016.

[19] C. Xie, O. Koyejo, and I. Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

[20] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

8 Appendix

8.1 Proofs

8.1.1 Preliminaries

We use the following lemma to bound the aggregated vectors.

Lemma 1. *(Bounded Score) Without loss of generality, we denote the $m - q$ correct elements in $\{\tilde{v}_i : i \in [m]\}$ as $\{v_i : i \in [m - q]\}$. Sorting the correct vectors by the online descendant score, we obtain $\{v_{(i)} : i \in [m - q]\}$. Thus, we have the following inequality:*

$$Score_{\gamma,\rho}(\tilde{v}_{(i)}, x) \geq Score_{\gamma,\rho}(v_{(i)}, x), \forall i \in [m - q],$$

or, by flipping the signs on both sides, it is equivalent to

$$f_r(x - \gamma \tilde{v}_{(i)}) - f_r(x) + \rho\|\tilde{v}_{(i)}\|^2 \leq f_r(x - \gamma v_{(i)}) - f_r(x) + \rho\|v_{(i)}\|^2, \forall i \in [m - q],$$

Proof. We prove the lemma by contradiction.

Assume that $Score_{\gamma,\rho}(\tilde{v}_{(i)}, x) < Score_{\gamma,\rho}(v_{(i)}, x)$. Thus, there are i correct vectors having larger scores than $\tilde{v}_{(i)}$. However, because $\tilde{v}_{(i)}$ is the ith element in $\{\tilde{v}_{(i)} : i \in [m]\}$, there should be at most $i - 1$ vectors having larger scores than it, which yields a contradiction. □

Lemma 2. *(Bounded Distance) Assume that $\mu\gamma + \frac{2\rho}{\gamma} \geq 1$. For $\forall i \in [m - q]$, we have $\frac{\sum_{i=1}^{m-q} \mathbb{E}\|\nabla f_r(x) - \tilde{v}_{(i)}\|^2}{m - q} \leq V_r + \left[\alpha\left(L\gamma + \frac{2\rho}{\gamma} - 1\right) + 1\right] V.$*

Proof. Note that all the expectations are taken on both z_i's (in Definition 2) and the correct vectors $\{v_i : i \in [m - q]\}$.

We first bound $\|\nabla f_r(x) - \tilde{v}_{(i)}\|^2 = \|\nabla f_r(x)\|^2 + \|\tilde{v}_{(i)}\|^2 - 2\langle\nabla f_r(x), \tilde{v}_{(i)}\rangle$.

Using Assumption 1 (lower bounded Taylor's approximation), taking $y = x - \gamma\tilde{v}_{(i)}$, we obtain

$$-\gamma\langle\nabla f_r(x), \tilde{v}_{(i)}\rangle \leq f_r(x - \gamma\tilde{v}_{(i)}) - f_r(x) - \frac{\mu\gamma^2}{2}\|\tilde{v}_{(i)}\|^2.$$

Using Lemma 1, and Assumption 1 (smoothness), we obtain

$$- \gamma\langle\nabla f_r(x), \tilde{v}_{(i)}\rangle$$
$$\leq f_r(x - \gamma\tilde{v}_{(i)}) - f_r(x) - \frac{\mu\gamma^2}{2}\|\tilde{v}_{(i)}\|^2 \quad \triangleright \text{lower bounded Taylor's approximation}$$
$$\leq f_r(x - \gamma v_{(i)}) - f_r(x) + \rho\|v_{(i)}\|^2 - \rho\|\tilde{v}_{(i)}\|^2 - \frac{\mu\gamma^2}{2}\|\tilde{v}_{(i)}\|^2 \quad \triangleright \text{Lemma 1}$$
$$\leq -\gamma\langle\nabla f_r(x), v_{(i)}\rangle + \frac{L\gamma^2}{2}\|v_{(i)}\|^2 + \rho\|v_{(i)}\|^2 - \rho\|\tilde{v}_{(i)}\|^2 - \frac{\mu\gamma^2}{2}\|\tilde{v}_{(i)}\|^2. \quad \triangleright \text{smoothness}$$

Thus, we have

$$\|\nabla f_r(x) - \tilde{v}_{(i)}\|^2$$
$$= \|\nabla f_r(x)\|^2 + \|\tilde{v}_{(i)}\|^2 - 2\langle\nabla f_r(x), \tilde{v}_{(i)}\rangle$$
$$\leq \|\nabla f_r(x)\|^2 + \|\tilde{v}_{(i)}\|^2 + \frac{2}{\gamma}\left[-\gamma\langle\nabla f_r(x), v_{(i)}\rangle + \frac{L\gamma^2}{2}\|v_{(i)}\|^2 + \rho\|v_{(i)}\|^2 - \rho\|\tilde{v}_{(i)}\|^2 - \frac{\mu\gamma^2}{2}\|\tilde{v}_{(i)}\|^2\right]$$
$$= \|\nabla f_r(x)\|^2 - 2\langle\nabla f_r(x), v_{(i)}\rangle + \left(L\gamma + \frac{2\rho}{\gamma}\right)\|v_{(i)}\|^2 + \left(1 - \frac{2\rho}{\gamma} - \mu\gamma\right)\|\tilde{v}_{(i)}\|^2$$
$$\leq \|\nabla f_r(x) - v_{(i)}\|^2 + \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)\|v_{(i)}\|^2. \quad \triangleright L\gamma + \frac{2\rho}{\gamma} \geq \mu\gamma + \frac{2\rho}{\gamma} \geq 1$$

10

By summing up, we have

$$\sum_{i=1}^{m-q} \mathbb{E}\|\nabla f_r(x) - \tilde{v}_{(i)}\|^2$$

$$\leq \sum_{i=1}^{m-q} \mathbb{E}\|\nabla f_r(x) - v_{(i)}\|^2 + \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)\mathbb{E}\|v_{(i)}\|^2$$

$$= \sum_{i=1}^{m-q} \mathbb{E}\|\nabla f_r(x) - v_i\|^2 + \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)\mathbb{E}\|v_i\|^2$$

$$= \sum_{i=1}^{m-q} \mathbb{E}\|\nabla f_r(x) - g\|^2 + \mathbb{E}\|g - v_i\|^2 + \left(L\gamma + \frac{2\rho}{\gamma} - 1\right)\mathbb{E}\|v_i\|^2$$

$$\leq (m-q)V_r + \left[\alpha\left(L\gamma + \frac{2\rho}{\gamma} - 1\right) + 1\right](m-q)V. \quad \triangleright \textit{Assumption 2}$$

$\square$

Note that we can take $\rho = \frac{\gamma}{2}$ to obtain a simpler form of the bound $\mathbb{E}\|\tilde{v}_{(i)} - g\|^2 \leq (\alpha L\gamma + 1)V$. For non-convex loss functions with $\mu < 0$, we can increase ρ so that the condition $\mu\gamma + \frac{2\rho}{\gamma} \geq 1$ can be satisfied.

Now we can use the lemma above to bound the aggregated gradient estimator, $\texttt{Zeno}_b(\cdot)$.

Theorem 1. *(Bounded Variance) Assume that q out of the m candidate gradient estimators $\{\tilde{v}_i : i \in [m]\}$ are Byzantine, with $q \leq b < m$, and x is the current value of the parameters. We further assume that ρ is large enough such that $\mu\gamma + \frac{2\rho}{\gamma} \geq 1$. By averaging the first $m-b$ vectors in $\{\tilde{v}_{(i)} : i \in [m]\}$, the aggregated vector $\texttt{Zeno}_b(\{\tilde{v}_i : i \in [m]\})$ is bounded by*

$$\mathbb{E}\left\|\texttt{Zeno}_b(\{\tilde{v}_i : i \in [m]\}) - \nabla F(x)\right\|^2 \leq \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2},$$

where $m > b \geq q$, and $\beta = \alpha\left(L\gamma + \frac{2\rho}{\gamma} - 1\right)$.

Proof. For convenience, we denote $g = \nabla F(x)$. Assume that in the $m-b$ selected gradient estimators, there are q_1 Byzantine vectors, where $q_1 \leq \min(q, m-b)$. Then, assuming that $m - b - q_1 > 0$, and $q_1 > 0$, we obtain the following bound:

$$\mathbb{E}\left\|\texttt{Zeno}_b(\{\tilde{v}_i : i \in [m]\}) - g\right\|^2$$

$$= \mathbb{E}\left\|\frac{1}{m-b}\sum_{i=1}^{n-b}(\tilde{v}_{(i)} - g)\right\|^2$$

$$= \mathbb{E}\left\|\frac{\sum_{\text{correct } i}(\tilde{v}_{(i)} - g)}{m-b} + \frac{\sum_{\text{Byzantine } i}(\tilde{v}_{(i)} - g)}{m-b}\right\|^2$$

$$\leq \frac{2(m-b-q_1)^2}{(m-b)^2}\mathbb{E}\left\|\frac{\sum_{\text{correct } i}(\tilde{v}_{(i)} - g)}{m-b-q_1}\right\|^2 + \frac{2q_1^2}{(m-b)^2}\mathbb{E}\left\|\frac{\sum_{\text{Byzantine } i}(\tilde{v}_{(i)} - g)}{q_1}\right\|^2.$$

11

Note that for arbitrary subset $\mathcal{S} \subseteq [m-q]$, $|\mathcal{S}| = m-b-q_1$, we have the following bound:

$$\mathbb{E}\left\|\frac{\sum_{i\in\mathcal{S}}(v_i-g)}{m-b-q_1}\right\|^2$$

$$= \mathbb{E}\left\|\frac{\sum_{i\in[m-q]}(v_i-g) - \sum_{i\notin\mathcal{S}}(v_i-g)}{m-b-q_1}\right\|^2$$

$$\leq 2\mathbb{E}\left\|\frac{\sum_{i\in[m-q]}(v_i-g)}{m-b-q_1}\right\|^2 + 2\mathbb{E}\left\|\frac{\sum_{i\notin\mathcal{S}}(v_i-g)}{m-b-q_1}\right\|^2$$

$$= \frac{2(m-q)^2}{(m-b-q_1)^2}\mathbb{E}\left\|\frac{\sum_{i\in[m-q]}(v_i-g)}{m-q}\right\|^2 + \frac{2(b-q+q_1)^2}{(m-b-q_1)^2}\mathbb{E}\left\|\frac{\sum_{i\notin\mathcal{S}}(v_i-g)}{b-q+q_1}\right\|^2$$

$$\leq \frac{2(m-q)^2}{(m-b-q_1)^2}\frac{V}{m-q} + \frac{2(b-q+q_1)^2}{(m-b-q_1)^2}\frac{\sum_{i\in[m-q]}\|v_i-g\|^2}{b-q+q_1}$$

$$\leq \frac{2(m-q)^2}{(m-b-q_1)^2}\frac{V}{m-q} + \frac{2(b-q+q_1)^2}{(m-b-q_1)^2}\frac{(m-q)V}{b-q+q_1}$$

$$= \frac{2(b-q+q_1+1)(m-q)V}{(m-b-q_1)^2}.$$

Thus, we obtain

$$\mathbb{E}\left\|\mathtt{Zeno}_b(\{\tilde{v}_i : i\in[m]\}) - g\right\|^2$$

$$\leq \frac{2(m-b-q_1)^2}{(m-b)^2}\mathbb{E}\left\|\frac{\sum_{\text{correct }i}(\tilde{v}_{(i)}-g)}{m-b-q_1}\right\|^2 + \frac{2q_1^2}{(m-b)^2}\mathbb{E}\left\|\frac{\sum_{\text{Byzantine }i}(\tilde{v}_{(i)}-g)}{q_1}\right\|^2$$

$$\leq \frac{2(m-b-q_1)^2}{(m-b)^2}\mathbb{E}\left\|\frac{\sum_{\text{correct }i}(\tilde{v}_{(i)}-g)}{m-b-q_1}\right\|^2 + \frac{2q_1^2}{(m-b)^2}\frac{\sum_{i=1}^{m-q}\mathbb{E}\|\tilde{v}_{(i)}-g\|^2}{q_1}$$

$$\leq \frac{2(m-b-q_1)^2}{(m-b)^2}\frac{2(b-q+q_1+1)(m-q)V}{(m-b-q_1)^2} + \frac{4q_1^2}{(m-b)^2}\frac{\sum_{i=1}^{m-q}\mathbb{E}\|\tilde{v}_{(i)}-\nabla f_r(x)\|^2 + \mathbb{E}\|g-\nabla f_r(x)\|^2}{q_1}$$

$$\leq \frac{4(b-q+q_1+1)(m-q)V}{(m-b)^2} + \frac{4q_1(m-q)}{(m-b)^2}\left[2V_r + \left[\alpha\left(L\gamma+\frac{2\rho}{\gamma}-1\right)+1\right]V\right]$$

$$\leq \frac{4(b+1)(m-q)V + 4q(m-q)(\beta+1)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2} \quad \triangleright q_1 \leq q$$

$$\leq \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}, \quad \triangleright q < b+1$$

where $\beta = \alpha\left(L\gamma + \frac{2\rho}{\gamma}-1\right)$.

It is easy to check that when $q_1 = 0$ or $m-b-q_1 = 0$, this upper bound still holds. $\qquad\square$

8.1.2 Strongly convex functions

The following lemma is from [3].

Lemma 3. *Let F be μ_F-strongly convex and L_F-smooth. Then for all $x, y \in \mathbb{R}^d$, one has*

$$\langle \nabla F(x) - \nabla F(y), x-y\rangle \geq \frac{\mu_F L_F}{\mu_F + L_F}\|x-y\|^2 + \frac{1}{\mu_F + L_F}\|\nabla F(x) - \nabla F(y)\|^2.$$

Theorem 2. *Assume that $F(x)$ is μ_F-strongly convex and L_F-smooth, where $0 < \mu_F \leq L_F$. We take $\gamma \leq \frac{2}{\mu_F + L_F}$. Using $\mathtt{Zeno}(\cdot)$ and taking the same assumptions in Theorem 1, we obtain linear convergence with a constant error after T iterations:*

$$\mathbb{E}\|x^T - x^*\| \leq \left(1 - \frac{\gamma\mu_F L_F}{\mu_F + L_F}\right)^T\|x^0 - x^*\| + \frac{\mu_F + L_F}{\mu_F L_F}\gamma\sqrt{\Delta},$$

where $\Delta = \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}$, $\beta = \alpha\left(L\gamma + \frac{2\rho}{\gamma}-1\right)$.

Proof. Denote that $g(x^t) = \texttt{Zeno}_b(\{g_i(x^t) : i \in [m]\})$, and $x^{t+1} = x^t - \gamma g(x^t)$.

Thus, we have

$$\|x^{t+1} - x^*\| = \|x^t - \gamma g(x^t) - x^*\| \le \|x^t - \gamma \nabla F(x^t) - x^*\| + \|\gamma \nabla F(x^t) - \gamma g(x^t)\|. \quad (2)$$

Furthermore, we have

$$\|x^t - \gamma \nabla F(x^t) - x^*\|^2 = \|x^t - x^*\|^2 + \gamma^2 \|\nabla F(x^t)\|^2 - 2\gamma \left\langle x^t - x^*, \nabla F(x^t) \right\rangle.$$

Using the μ_F-strong convexity, L_F-smoothness of $F(x)$, and Lemma 3 with $x = x^t$, $y = x^*$, we have

$$\left\langle \nabla F(x^t), x^t - x^* \right\rangle \ge \frac{\mu_F L_F}{\mu_F + L_F} \|x^t - x^*\|^2 + \frac{1}{\mu_F + L_F} \|\nabla F(x^t)\|^2.$$

Taking $\gamma \le \frac{2}{\mu_F + L_F}$, we obtain

$$\|x^t - \gamma \nabla F(x^t) - x^*\|^2$$
$$\le \|x^t - x^*\|^2 + \gamma^2 \|\nabla F(x^t)\|^2 - 2\gamma \left(\frac{\mu_F L_F}{\mu_F + L_F} \|x^t - x^*\|^2 + \frac{1}{\mu_F + L_F} \|\nabla F(x^t)\|^2 \right)$$
$$\le \left(1 - \frac{2\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\|^2 + \gamma \left(\gamma - \frac{2}{\mu_F + L_F} \right) \|\nabla F(x^t)\|^2$$
$$\le \left(1 - \frac{2\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\|^2.$$

Note that when $\gamma \le \frac{2}{\mu_F + L_F}$, we have $\frac{2\gamma \mu_F L_F}{\mu_F + L_F} \le \frac{4\mu_F L_F}{(\mu_F + L_F)^2} \le 1$. Using $\sqrt{1-a} \le 1 - \frac{a}{2}, \forall a \le 1$, we obtain

$$\|x^t - \gamma \nabla F(x^t) - x^*\| \le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\|.$$

Combined with Equation 2, we obtain

$$\|x^{t+1} - x^*\| \le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\| + \gamma \|\nabla F(x^t) - g(x^t)\|.$$

Conditional on x^t, taking the expectation on both sides, we obtain

$$\mathbb{E}\|x^{t+1} - x^*\|$$
$$\le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\| + \gamma \mathbb{E}\|\nabla F(x^t) - g(x^t)\|$$
$$\le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\| + \gamma \sqrt{\mathbb{E}\|\nabla F(x^t) - g(x^t)\|^2} \quad \triangleright \textit{Jenssen's inequality}$$
$$\le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right) \|x^t - x^*\| + \gamma \sqrt{\Delta}, \quad \triangleright \textit{Theorem 1}$$

where $\Delta = \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}$.

By telescoping and taking total expectation, we obtain

$$\mathbb{E}\|x^T - x^*\|$$
$$\le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right)^T \|x^0 - x^*\| + \sum_{i=0}^{T-1} \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right)^i \gamma \sqrt{\Delta}$$
$$\le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right)^T \|x^0 - x^*\| + \sum_{i=0}^{+\infty} \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right)^i \gamma \sqrt{\Delta}$$
$$\le \left(1 - \frac{\gamma \mu_F L_F}{\mu_F + L_F} \right)^T \|x^0 - x^*\| + \frac{\mu_F + L_F}{\mu_F L_F} \gamma \sqrt{\Delta}. \quad \triangleright \sum_{i=0}^{+\infty} a^i = \frac{1}{1-a}, \textit{for } |a| < 1$$

$\square$

8.1.3 General functions

For general non-strongly convex functions and non-convex functions, we provide the following convergence guarantees.

Theorem 3. *Assume that $F(x)$ is L_F-smooth and potentially non-convex, where $0 < L_F$. We take $\gamma \leq \frac{1}{L_F}$. Using $\mathtt{Zeno}(\cdot)$ and taking the same assumptions in Theorem 1, we obtain the convergence with a constant error after T iterations:*

$$\frac{\sum_{i=0}^{T-1} \mathbb{E}\|\nabla F(x^i)\|^2}{T} \leq \frac{2}{\gamma T} \left[F(x^0) - F(x^*) \right] + \Delta,$$

where $\Delta = \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}$, $\beta = \alpha \left(L\gamma + \frac{2\rho}{\gamma} - 1 \right)$.

Proof. Denote that $g(x^t) = \mathtt{Zeno}_b(\{g_i(x^t) : i \in [m]\})$, and $x^{t+1} = x^t - \gamma g(x^t)$.

To prove the convergence with constant error, we first bound the descendant of the loss value in each iteration.

$$
\begin{aligned}
F(x^{t+1}) &\leq F(x^t) + \langle \nabla F(x^t), x^{t+1} - x^t \rangle + \frac{L_F}{2}\|x^{t+1} - x^t\|^2 \quad \triangleright \textit{smoothness} \\
&= F(x^t) - \gamma \langle \nabla F(x^t), g(x^t) \rangle + \frac{L_F \gamma^2}{2}\|g(x^t)\|^2 \\
&\leq F(x^t) - \gamma \langle \nabla F(x^t), g(x^t) \rangle + \frac{\gamma}{2}\|g(x^t)\|^2 \quad \triangleright \gamma \leq \frac{1}{L_F} \\
&= F(x^t) - \frac{\gamma}{2}\|\nabla F(x^t)\|^2 + \frac{\gamma}{2}\|\nabla F(x^t) - g(x^t)\|^2.
\end{aligned}
$$

Thus, we obtain

$$F(x^{t+1}) - F(x^*) \leq F(x^t) - F(x^*) - \frac{\gamma}{2}\|\nabla F(x^t)\|^2 + \frac{\gamma}{2}\|\nabla F(x^t) - g(x^t)\|^2.$$

By telescoping and taking total expectation, we obtain

$$0 \leq \mathbb{E}\left[F(x^T) - F(x^*)\right] \leq F(x^0) - F(x^*) - \frac{\gamma}{2} \sum_{i=0}^{T-1} \mathbb{E}\|\nabla F(x^i)\|^2 + \frac{\gamma}{2}T\Delta,$$

where $\Delta = \frac{4(b+1)(m-q)(\beta+2)V}{(m-b)^2} + \frac{8q(m-q)\beta V_r}{(m-b)^2}$.

By rearranging the terms, we obtain the desired result

$$\frac{\sum_{i=0}^{T-1} \mathbb{E}\|\nabla F(x^i)\|^2}{T} \leq \frac{2}{\gamma T} \left[F(x^0) - F(x^*) \right] + \Delta.$$

$\square$

8.2 Experimental Details

In Table 2 and 3, we show the detailed network structures of the MLP and CNN used in our experiments.

8.3 Additional Experiments

In this section, we illustrate the additional empirical results.

In Figure 5 and 6, we show the experiments on MNIST dataset with softmax regression. The results are similar to the experiments on MNIST dataset with MLP.

In Figure 7 and 8, we show the experiments on CIFAR-10 dataset with CNN. The complexity of the neural networks structure and the dataset enlarge the variance of the gradients, which results worse convergence when there are Byzantine failures. In most cases, $\mathtt{Zeno}$ outperforms the baselines. The only exception is in Figure 7(b). However, we can tune n_r and ρ to improve the performance of $\mathtt{Zeno}$.

Table 2: MLP Summary

Layer (type)	Parameters	Previous Layer
flatten(Flatten)	null	data
fc1(FullyConnected)	#output=128	flatten
relu1(Activation)	null	fc1
fc2(FullyConnected)	#output=128	relu1
relu2(Activation)	null	fc2
fc3(FullyConnected)	#output=10	relu2
softmax(SoftmaxOutput)	null	fc3

Table 3: CNN Summary

Layer (type)	Parameters	Previous Layer
conv1(Convolution)	channels=32, kernel_size=3, padding=1	data
activation1(Activation)	null	conv1
conv2(Convolution)	channels=32, kernel_size=3, padding=1	activation1
activation2(Activation)	null	conv2
pooling1(Pooling)	pool_size=2	activation2
dropout1(Dropout)	probability=0.2	pooling1
conv3(Convolution)	channels=64, kernel_size=3, padding=1	dropout1
activation2(Activation)	null	conv3
conv4(Convolution)	channels=64, kernel_size=3, padding=1	activation2
activation4(Activation)	null	conv4
pooling2(Pooling)	pool_size=2	activation4
dropout2(Dropout)	probability=0.2	pooling2
flatten1(Flatten)	null	dropout2
fc1(FullyConnected)	#output=1024	flatten1
activation5(Activation)	null	fc1
dropout3(Dropout)	probability=0.2	activation5
fc2(FullyConnected)	#output=1024	dropout3
activation6(Activation)	null	fc2
dropout4(Dropout)	probability=0.2	activation6
fc3(FullyConnected)	#output=10	dropout4
softmax(SoftmaxOutput)	null	fc3

Another practical concern is that, to execute Zeno, the sever must be able to draw samples from the entire training dataset, which may not be possible due to some privacy-preserving policies. In practice, Zeno can use the samples drawn from an independent validation dataset instead of the training set. In the following experiments, we test the performance of a variant of Zeno, which draws samples from the test set instead of the training set to compute the stochastic descendant scores. Such variant is referred to as Zeno with test set. In Figure 9 and 10, we show the accuracy on MNIST test set with MLP. In Figure 11 and 12, we show the accuracy on MNIST training set with MLP. We can see that using the test set for Zeno will still result in good convergence. In some cases, Zeno with sampling from the test set show better performance than averaging without Byzantine failures. For the training accuracy, Zeno shows similar performance, no matter the samples for the stochastic descendant score are drawn from the test set or the training set.
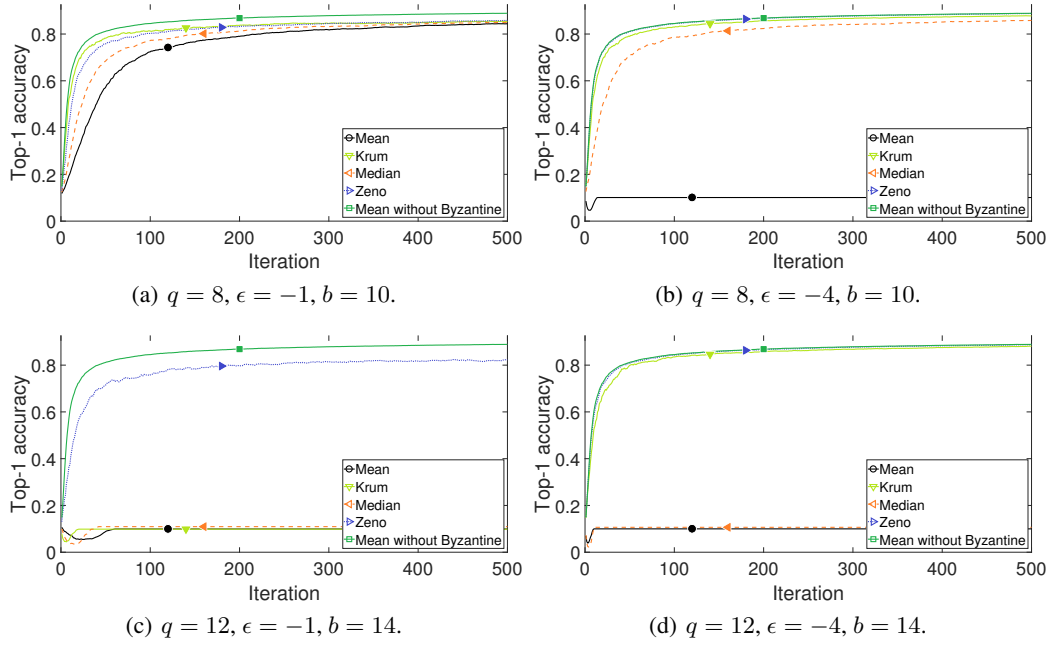
Figure 5: Top-1 accuracy of softmax regression on MNIST with sign-flipping attack and different hyperparameters. In all the 4 figures, $\gamma = 0.05$, $\rho = \frac{\gamma}{20}$, worker batch size is 32, Zeno batch size is 4. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.
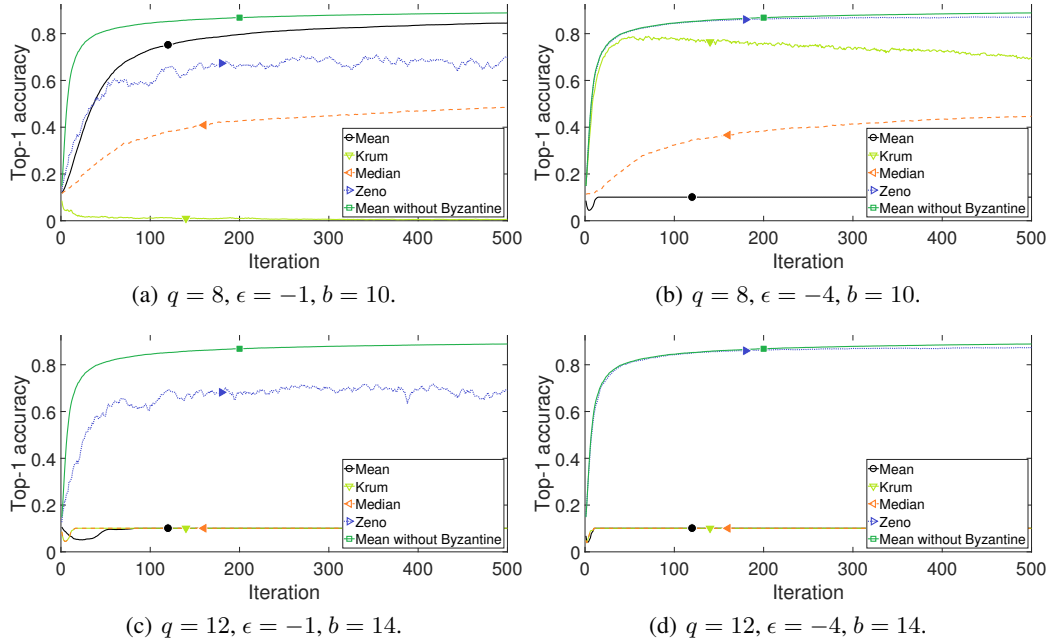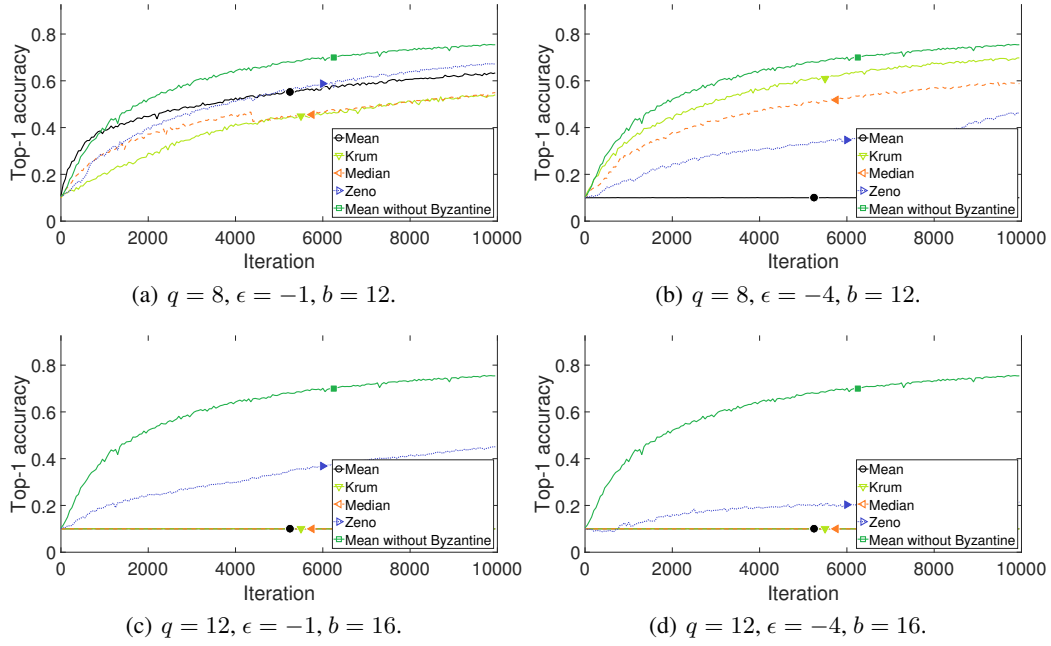

Figure 6: Top-1 accuracy of softmax regression on MNIST with omniscient attack and different hyperparameters. In all the 4 figures, $\gamma = 0.05$, $\rho = \frac{\gamma}{20}$, worker batch size is 32, Zeno batch size is 4. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.

Figure 7: Top-1 accuracy of CNN on CIFAR-10 with sign-flipping attack and different hyperparameters. In all the 4 figures, $\gamma = 0.005$, $\rho = \gamma e^{-6}$, worker batch size is 64, Zeno batch size is $n_r = 64$. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.
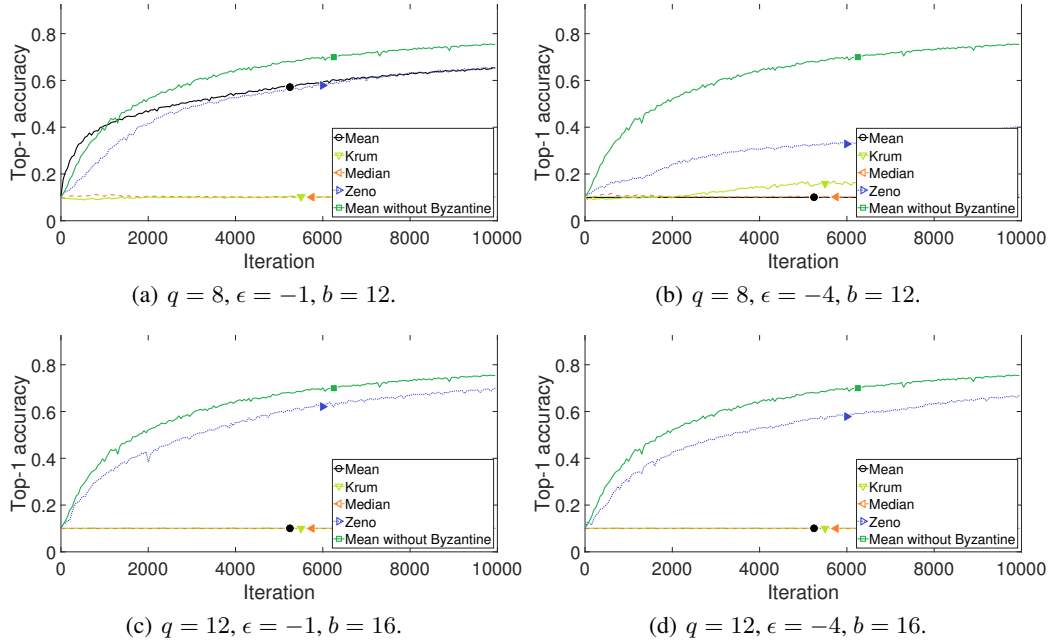

Figure 8: Top-1 accuracy of CNN on CIFAR-10 with omniscient attack and different hyperparameters. In all the 4 figures, $\gamma = 0.005$, $\rho = \gamma e^{-6}$, worker batch size is 64, Zeno batch size is $n_r = 64$. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.

17

(a) $q = 8, \epsilon = -1, b = 10$.

(b) $q = 8, \epsilon = -4, b = 10$.

(c) $q = 12, \epsilon = -1, b = 14$.

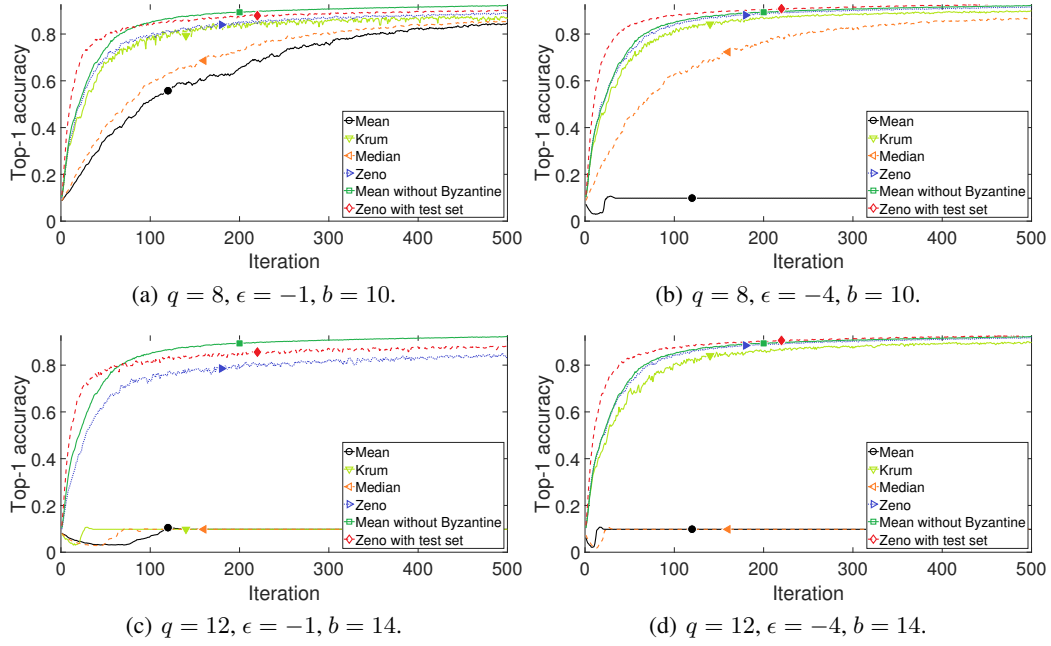(d) $q = 12, \epsilon = -4, b = 14$.

Figure 9: Top-1 accuracy of MLP on MNIST test set with sign-flipping attack and different hyper-parameters. $\gamma = 0.1$, worker batch size is 32. For Zeno, $\rho = \frac{\gamma}{40}$, $n_r = 12$. For Zeno with test set, $\rho = \frac{\gamma}{100}$, $n_r = 16$. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.

(a) $q = 8, \epsilon = -1, b = 10$.

(b) $q = 8, \epsilon = -4, b = 10$.

(c) $q = 12, \epsilon = -1, b = 14$.
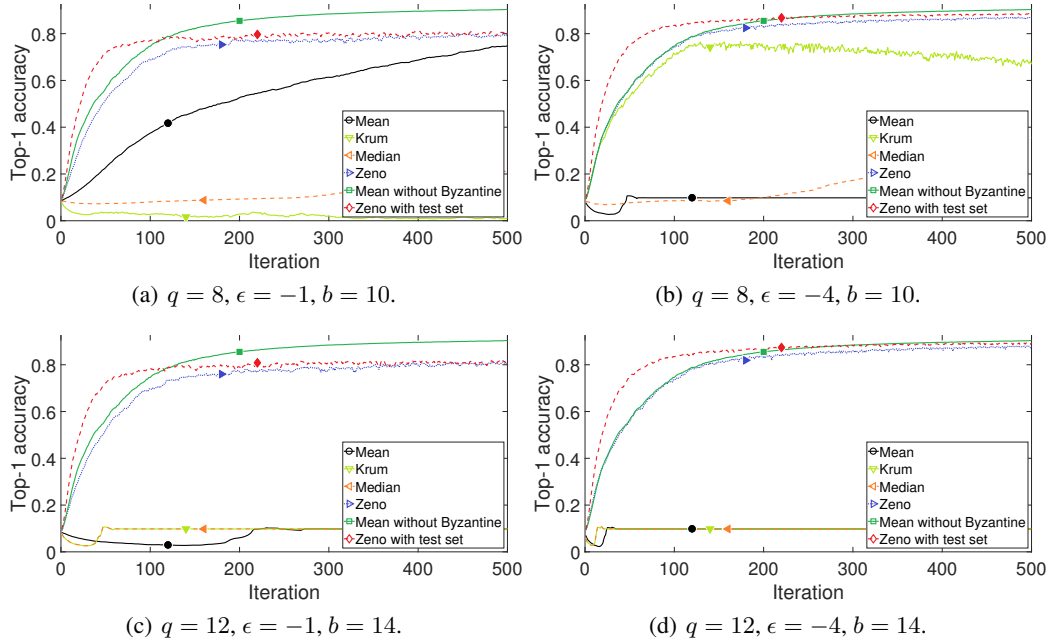
(d) $q = 12, \epsilon = -4, b = 14$.

Figure 10: Top-1 accuracy of MLP on MNIST test set with omniscient attack and different hyperpa-rameters. $\gamma = 0.05$, worker batch size is 32. For Zeno, $\rho = \frac{\gamma}{100}$, $n_r = 12$. For Zeno with test set, $\rho = \frac{\gamma}{100}$, $n_r = 16$. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.
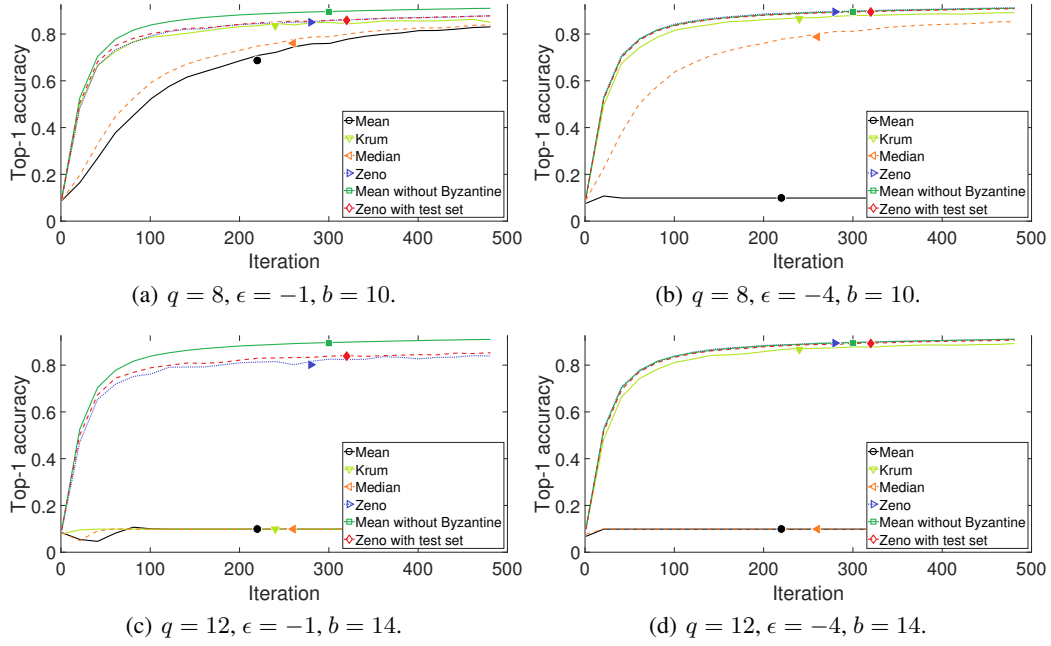
Figure 11: Top-1 accuracy of MLP on MNIST training set with sign-flipping attack and different hyperparameters. $\gamma = 0.1$, worker batch size is 32. For Zeno, $\rho = \frac{\gamma}{40}$, $n_r = 12$. For Zeno with test set, $\rho = \frac{\gamma}{100}$, $n_r = 16$. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.
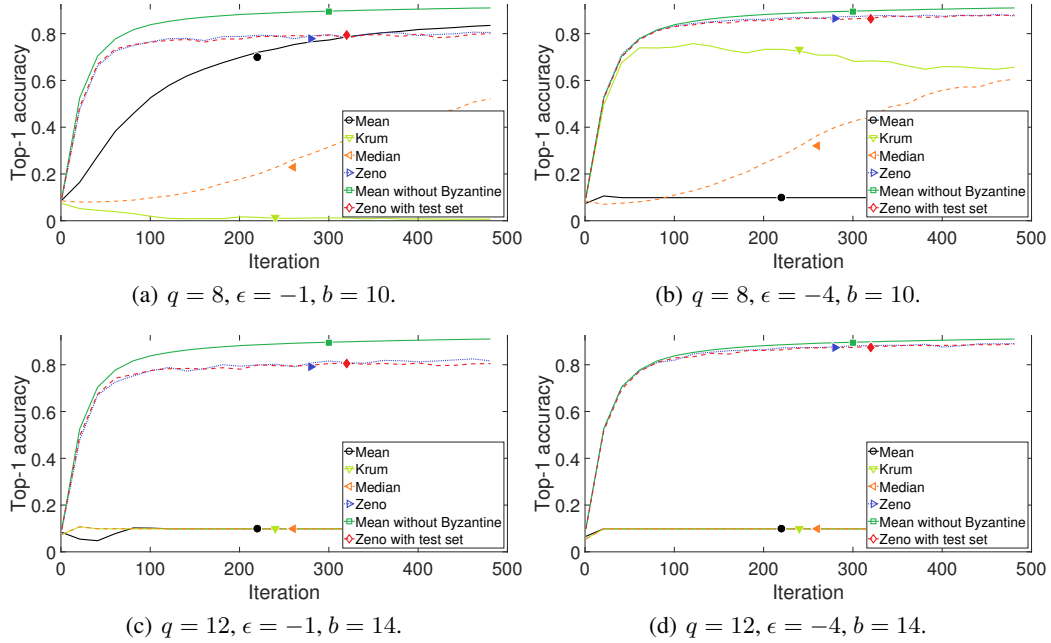

Figure 12: Top-1 accuracy of MLP on MNIST training set with omniscient attack and different hyperparameters. $\gamma = 0.05$, worker batch size is 32. For Zeno, $\rho = \frac{\gamma}{100}$, $n_r = 12$. For Zeno with test set, $\rho = \frac{\gamma}{100}$, $n_r = 16$. (a,b) and (c,d) differs in q. (a,c) and (b,d) differs in ϵ.